Lista Duplamente Encadeada

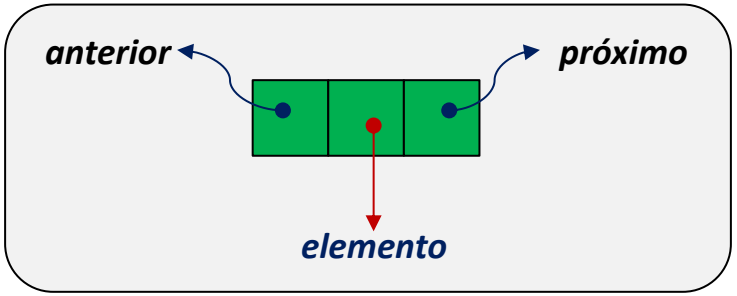
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca
CEFET-RJ

Lista Simplesmente Encadeada

- **Desvantagens:**
  - **Como retornar ao nó anterior?**
    - **O nó contém apenas uma referência para o próximo nó.**
      - **Uma solução é ter referência ao nó anterior e ao próximo nó – lista duplamente encadeada**
  - **Como voltar para o início ao atingir o último nó da lista?**
    - **Uma solução é ligar o último nó ao primeiro nó – lista circular**
      - **Pode ser simplesmente encadeada ou duplamente encadeada**
  - **Dependendo da forma como o algoritmo é construído, para realizar uma busca de um elemento, a lista deverá ser percorrida do início ao fim – O(n).**
    - **Árvore**

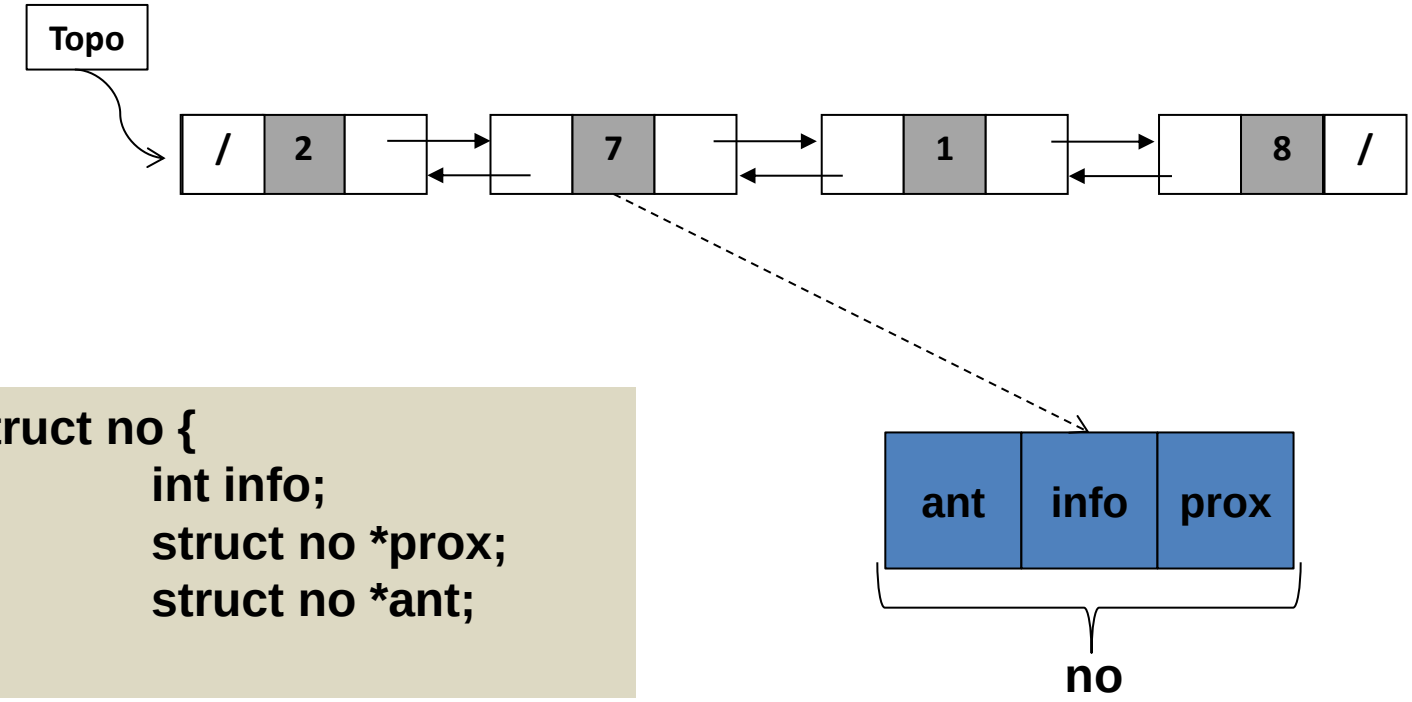
Lista Duplamente Encadeada

- **Extensão da lista simplesmente encadeada.**
  - **Pode-se percorrer nos dois sentidos da lista**
  - **Cada elemento da lista contém duas referências (dois ponteiros): uma para o nó anterior e outra para o próximo nó.**
  - **Podem ser lineares ou circulares.**



Lista Duplamente Encadeada

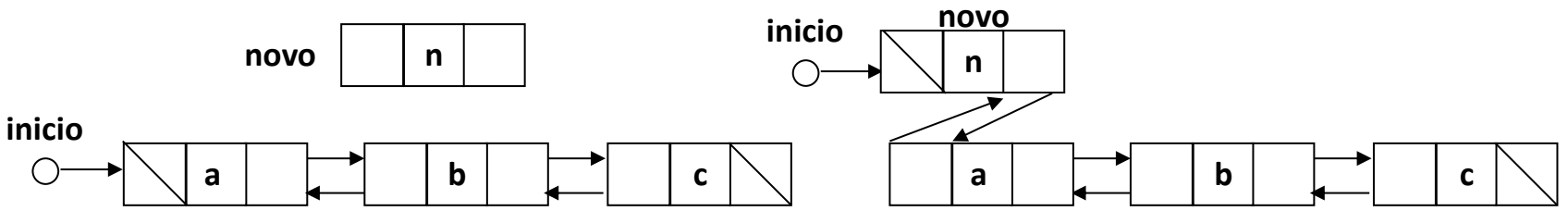
- **O primeiro e o último elemento da lista não possuem ligação - referência é nula.**



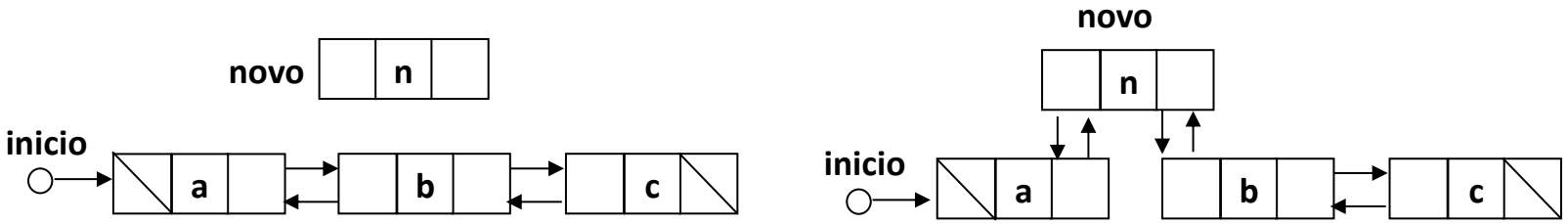

```
struct no {
        int info;
        struct no *prox;
        struct no *ant;
};
```

Inserção de elementos

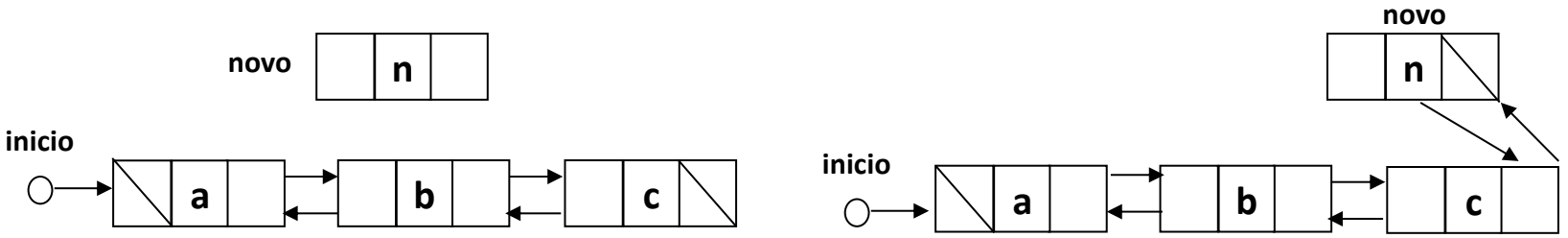
Inserindo um novo elemento na primeira posição



Inserindo um novo elemento entre nós

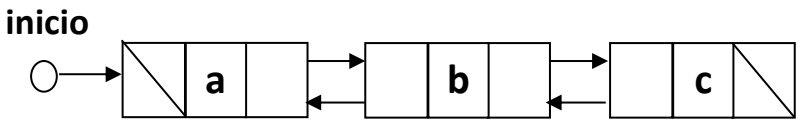


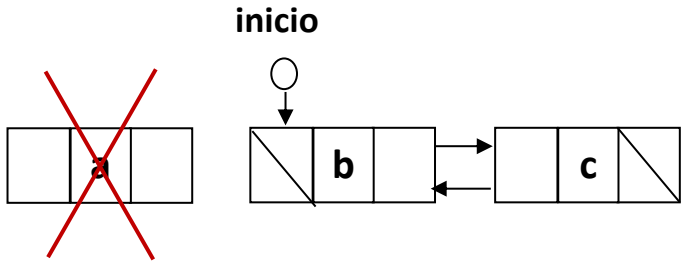
Inserindo um novo elemento na última posição



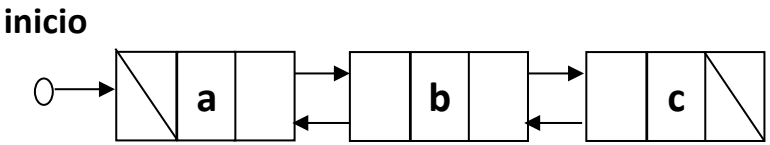
Retirada de elementos

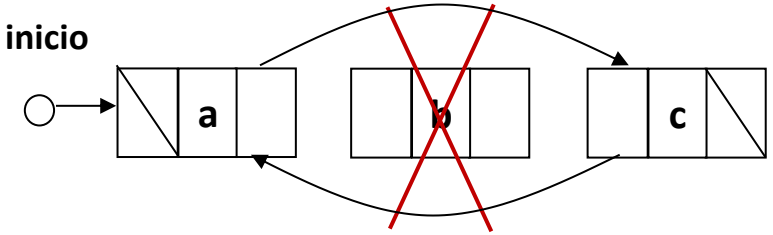
Removendo o primeiro elemento



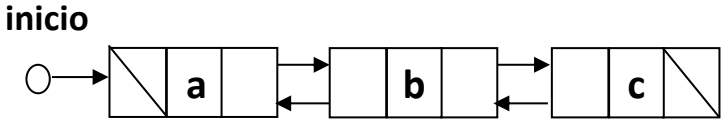


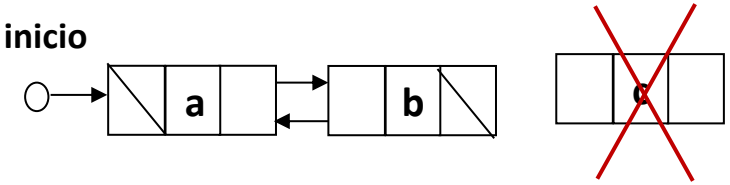
Removendo entre elementos





Removendo o último elemento





Exemplo - 1º Caso

```
#include <iostream>

using namespace std;

struct no {
        int info;
        struct no *ant;
        struct no *prox;
};

typedef struct no *noPtr;

void inserir(noPtr *);
void retirar(noPtr *);
void listar(noPtr);
int listaVazia(noPtr);
```

Exemplo – continuação

```
main()
{
   int op;
   noPtr inicio = NULL;
   do {
            cout  << "\n1: Inserir elemento na lista"
                  <<"\n2: Retirar elemento da lista"
                  << "\n3: Listar elementos"
                  << "\n0: Sair"
                  << "\n\nDigite a opcao (0 - 3): ";
            cin >> op;
            switch (op)
            {
                        case 1: inserir(&inicio); break;
                        case 2: retirar(&inicio); break;
                        case 3: listar(inicio); break;
            }
  } while (op != 0);
}
```

Exemplo – continuação

```
void inserir (noPtr * i) {

   noPtr p = new no;

   cout << "\nDigite o valor do elemento: ";
   cin >>   p->info;
    if (listaVazia(*i))
    {
        *i = p;
        p->prox = NULL;
        p->ant = NULL;
     }
     else  {
        p->ant= NULL;
        p->prox = *i;
        (*i)->ant = p;
        *i = p;
    }
}
```

Exemplo – continuação

```
void retirar (noPtr * i)
{
    noPtr p = *i;
    if (!listaVazia(*i))
    {
        if (p->prox == NULL)
              *i = NULL;
        else
        {
              *i = p->prox;   // *i = *i->prox
              (*i)->ant = NULL;  //p->prox->ant
         }
         delete(p);
         cout << "\nO elemento foi retirado da lista!\n";
    }
    else cout << "\n\nLista está Vazia!\n";
}
```

Exemplo – continuação

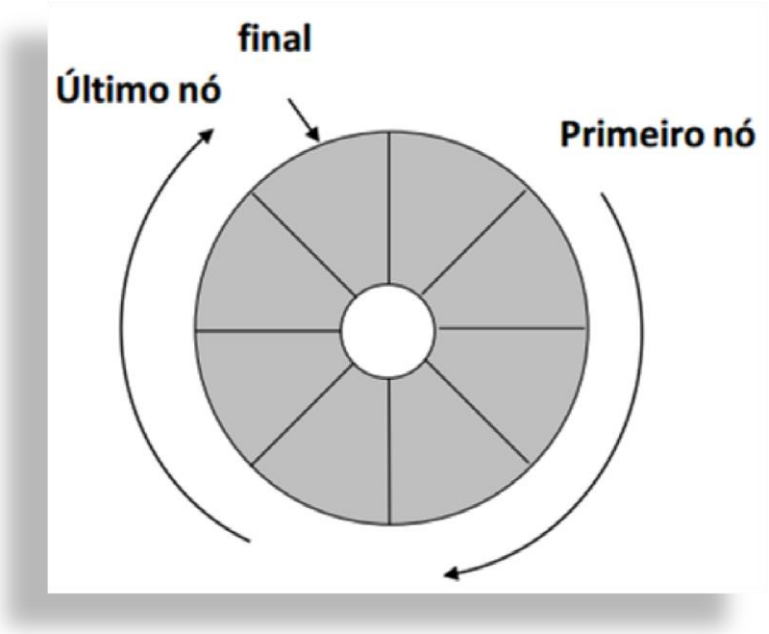
```
void listar(noPtr p)
{
    if (!listaVazia(p))
    {
            cout << "\nElementos da lista : \n";
            cout << "INICIO";
            while (p != NULL)
            {
                    cout << " --> " << p->info;
                    p = p->prox;
            }
            cout << " -- > NULL";
            }
    else
            cout << "\n\nLista está Vazia!\n";
}
```

Lista Circular

Centro Federal de Educação Tecnológica Celso Suckow da Fonseca
CEFET-RJ

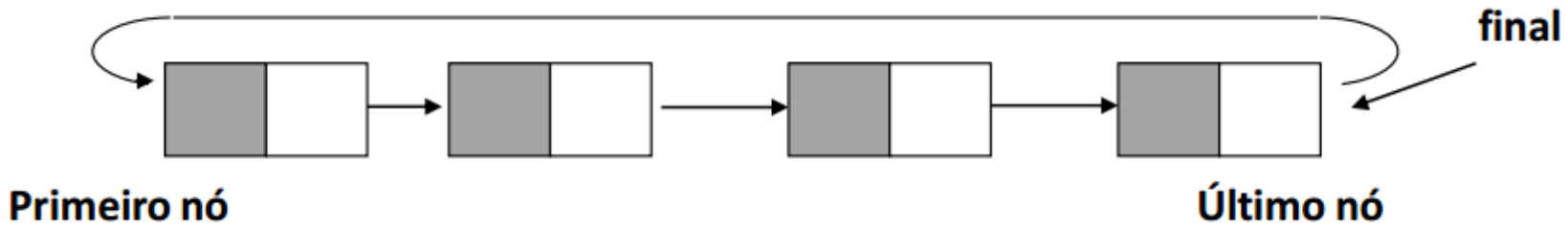
Lista Circular

- **A lista circular possui como endereço do último nó o primeiro nó da lista, em vez de conter um valor nulo.**
  - **Não é necessário percorrer todos os nós para sair do fim e retornar ao início.**
  - **Pode-se utilizar um ponteiro para apontar para o último (ou primeiro) nó da lista circular.**
- **A inclusão ou remoção de um elemento pode ser realizada tanto no <u>início</u> como no <u>final</u> da lista.**

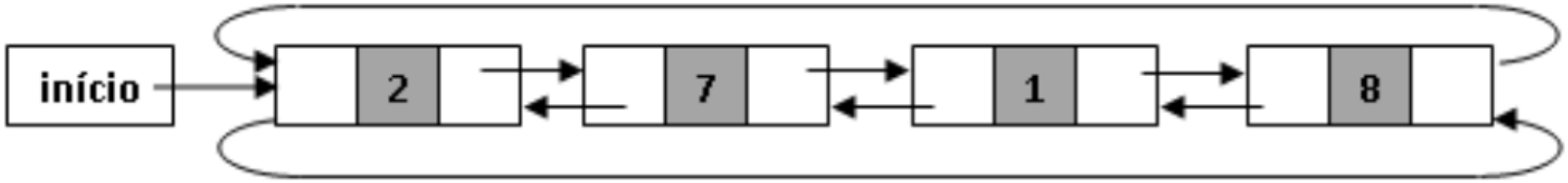


Lista circular

- **Simplesmente encadeada**
  - **O último nó aponta para o primeiro ou vice-versa.**



- **Duplamente encadeada**
  - **O primeiro elemento da lista aponta para o último elemento e vice-versa.**



Exemplo – Lista Circular

```
#include <iostream>

using namespace  std;

struct  no {
   int info;
   struct  no *prox;
};

typedef  struct no *noPtr;

int insere(noPtr *, int *);
void retira(noPtr *, int *);
void listar(noPtr, int);
bool listaVazia(noPtr);
int menu();
```

Exemplo – Lista Circular

```
main() {
  int op, qtde = 0;
  noPtr inicio = NULL;
  do {
          op = menu();
          switch (op) {
            case 1: qtde = insere(&inicio, &qtde);    //redundância
                    cout << "\nA lista possui " << qtde << " no(s).\n\n"; break;
            case 2: retira(&inicio, &qtde); break;
            case 3: listar(inicio, qtde); break;
          }
  } while  (op != 0);
}
```

Exemplo – Lista Circular

```
int  insere (noPtr * i, int * q) {                    //É uma lista circular??? Como fazer?
   noPtr  p = new no;
   cout << "\nDigite o valor do elemento: ";
   cin >> p->info;
   if (listaVazia(*i))
   {
      *i = p;
      p->prox = *i;
   } else {
      p->prox = *i;
      *i = p;
   }
   *q = *q + 1;                                       Redundância
   return *q;
}
```

Exemplo – Lista Circular

```
void retira (noPtr * i, int * q) {        //Mantém a lista circular??? O que fazer?
  noPtr p = *i;
  if (!listaVazia(*i))
  {
                    if (*q == 1)
                    {
                              *i = NULL;
                    }
                    else
                    *i = p->prox;  //*i = (*i)->prox
    delete(p);
    cout << "\nO elemento foi retirado da lista!\n";
    *q = *q - 1;
   } else  cout << "\n\nLista Vazia!\n";
}
```

Exemplo – Lista Circular

```
void listar(noPtr i, int q)
{
   if (!listaVazia(i))
  {
      for (int j = 0; j<q; j++)
     {
              cout << i->info << "\t";
              i = i->prox;
     }
  }
  else
     cout << "\n\nLista vazia!";

}
```

```
bool listaVazia(noPtr i)
{
              if(i)
                 return false;
              else
                 return true;
}
```

Exercícios

1. **Implementar as funções:**
   - **a. Buscar um elemento e fazer a sua retirada da lista duplamente encadeada.**
   - **b. Inserir e Retirar no final da lista duplamente encadeada.**
   - **c. Inserir os elementos em ordem decrescente numa lista duplamente encadeada.**
   - **d. Consultar determinado elemento na lista duplamente encadeada.**

2. **Sendo L1 e L2 ponteiros para duas listas duplamente encadeadas de números inteiros, implemente as seguintes funções:**
   - **a. Soma (L1) – somar os valores da lista L1 e apresentar o resultado.**
   - **b. Soma (L2) – somar os valores da lista L2 e apresentar o resultado.**
   - **c. Maior (L1, L2) – informar qual é o maior elemento e em qual lista ele se encontra.**
   - **d. Intercalação (L1, L2) – criar uma lista simplesmente encadeada (L3) que contenha os nós das listas L1 e L2 intercalados.**
   - **e. Transformar a lista L3 (item d) numa lista circular.**

Referências

- **Moraes. *Estruturas de Dados e Algoritmos – uma abordagem didática*. Ed. Futura**
- **Markenzon e Szwarcfiter. *Estruturas de Dados e seus Algoritmos*. Ed. LTC**
- **Deitel. Como Programar em C/C++. Ed. Pearson**